Interpolating Coastal Shallows via Trends in Terrestrial and Marine Topography

27. May 2018

Submitted by:
Aaron Mau
School of Oceanography, University of Washington
aaronmau@uw.edu

Submitted to:
Miles Logsdon
Ocean Technology Center, University of Washington
mlog@uw.edu

Acknowledgements

I would like to thank the students and faculty of UW Oceanography for organizing and realizing an insightful senior cruise through Hood Canal. In particular to faculty, I'd like to give a special thanks to my advisors Miles Logsdon, Arthur Nowell and Susanna Mitchell, without whom I'd have struggled to develop my thesis cohesively. I'd also like to extend my thanks to Kevin Pszczola, who processed some multibeam samples for my use in Winter Quarter. Thanks are also due to Dave Thorson, the driver of the *Welander* for single-beam data acquisition. Further thanks goes to Andrea Ogsdon, Emily Roland, Brian de Silva, and Parker MacCready for making time to sit down and discuss real-world implications and methodologies.

Executive Summary

Acquiring bathymetry data of coastal shallows in Puget Sound is both an important and difficult task. Puget Sound estuaries contain a large number of processes that dynamically impact coastal topography and communities, however modeling the influences of these processes is difficult without bathymetry data.

For this project, the processes of sediment deposition and wave action are explored by developing profiles of two common Puget Sound environments; river deltas and bluff-backed beaches. Single-beam transects of Mukilteo and Duckabush (Hood Canal) coastal estuaries were collected in January and February of 2018 using a Garmin 441s fishfinder with a 77/200 kHz variable transducer. These transects were then decomposed into bathymetric profiles perpendicular to the coastline, aligned to supplementing GMRT land topography and marine bathymetry available in the GeoMapApp visualization tool. GMRT data was used for assessing 4 different mathematical methods to interpolate the shallows; a linear interpolation, a cubic spline, a non-modified pchip, and a non-linear logistic regression.

The results from this project suggest that the most effective model between terrestrial elevation and marine bathymetry datasets is a simple pchip method, which most effectively preserves not only profile area relative to the observed data, but also demonstrates similar curvature.

Introduction

Seventy percent of the Earth's surface is ocean, 95% of which remains unexplored or unmapped in high resolution (NOAA, 2009). A considerable portion of unexplored ocean is at a depth typically not exceeding 100 m near coastlines - a data gap of missing bathymetry unofficially referred to as the "White Zone" due to how the lack of data is represented on maps. In these White Zone regions, a large number of natural and anthropogenic processes are known to take place and are arguably the most important oceanic region on Earth. Studies show incredible growth in size and population density of coastal cities, therefore making the White Zone a key region on anthropogenic changes; be it seawall construction or pollution (Neumann et al., 2015).

The White Zone is a difficult region to sample for many reasons; there are many methods each with their own problems. Perhaps the oldest method of bathymetry acquisition is the use of a lead line. Though this approach is an accurate, hands-on measurement, it is very time consuming. More modern sonar operations close to the shoreline can be conducted much faster, but are dangerous on larger survey vessels required for large transducers - especially for a multibeam swath (Figure 9). In multibeam bathymetry acquisition, operations at a shallower depth also reduce sampling effectiveness, as the swath under the ship shrinks as the seafloor shoals. White Zone can also be sampled by satellite to cover coastal areas, however errors may arise in the satellite data on the order of several meters which appear only more exaggerated in a shallow environment (Said et al., 2017). Emerging technologies in autonomous or remote controlled sensing appear capable of efficiently capturing seafloor topography. Unfortunately, these methods can cost tens of thousands of dollars while also presenting unique difficulties in deployment from a large, conventional vessel for a continuous map of the shallows to deeper waters (Hampson et al., 2011). So not only are these new methods inaccessible to smaller organizations or universities, but they may be difficult to implement for those with the funds for them.

The use of a mathematical model could be helpful to perceptualize the White Zone for two characteristic environments of Puget Sound shallows; bluff-backed beaches and river deltas. Given the availability of terrestrial topography and marine bathymetry data, the model functions to interpolate a profile between the data sets representing the White Zone at bluffs and deltas. A good model will cost significantly less to acquire or operate, and given the data it would need to interpolate the White Zone, could act in real-time on a survey vessel.

Background

Puget Sound boasts many kinds of different coastal environments from a complicated history of glaciation and anthropogenic change. Two particularly extensive environments within Puget Sound are bluff-backed beaches and river deltas; which contrast greatly in both a bathymetric profile and in the known forcings upon them (Shipman, 2008). Bluffs are formed from glacial carving and maintained by wave action, being more common in areas of higher wave activity such as the Puget Sound Main Basin. Due to more wave activity seafloor sediments are likely to be kicked up and transported away, resulting in a steeper slope profile. The steeper a profile gets, the more likely a landslide will be - something that can be identified at the deeper locations on a profile. Deltas are common in Puget Sound because they have numerous sources,

lakes and glaciers. Sediment slowly accumulates over time depending on the amount of water flux and sediment content carried to the mouth. Deltas tend to exist in lower wave activity areas, which enables sediment to accumulate and therefore developing a characteristically wider shallow region with fewer landslide events at greater depth.

A typical slope from land to sea consists of three regions; terrestrial, the White Zone, and then the marine floor. The terrestrial area is the only portion of the profile that is above water and most reliably available from satellite acquisition. Next is the shallow White Zone, which is difficult to sample. The White Zone itself consists of a shelf and steep slope. The shelf continues until the shelf break, a point where the profile transitions into a slope - often the steepest portion of the profile and where multibeam bathymetry data is most likely to overlap. Finally, the marine floor is the region that is most easily sampled by multibeam accurately, characterized by a gradual change from the White Zone slope into another shelf-like area.

The terrestrial and marine regions are most available for land acquisition from satellite or multibeam bathymetry, however conventional methods struggle with the White Zone for a number of reasons. The air-sea interface presents problems to satellites, therefore not capturing the shallows particularly well. For a multibeam transducer array, a ship cannot get too close to the shoreline without endangering the vessel, and the returns on the edges of the swath are often the least accurate. At an estuary, vessels will need to frequently account for changes in density as they traverse freshwater supplies. Density can influence speed of sound through the water column, such that a multibeam swath can "smile" or "frown" on the edges with an old sound velocity profile (Figure 10).

To numerically generate profiles of the White Zone, there are many different interpolation methods that can be used between available satellite and multibeam datasets. First is a *linear* fit, which functions entirely point-to-point. Linear fits are powerful because they can ignore a lot of noise and illustrate a trend clearly, however they are one dimensional and therefore don't capture any form of curvature.

Next are *splines*, which operate by forming a set of third-order polynomials from provided data points (Weisstein). Splines can, unlike a linear fit, introduce curvature between data points but can have interesting results if data points are not uniformly spaced on the horizontal or x axis. A spline will function based on the values provided and will use any coefficients necessary to connect three points. Problems arise when values are not equally spaced due to the relationships between sequential spline functions; a spline interpolation is a set of functions tangential to one another.

A way to accommodate for non-uniform spacing is to define a specified second derivative for spline endpoints, thereby preventing a growth in spacing from amplifying the curvature of the spline function. In a way, it's limiting the interaction between functions and defining how they should interact based on the overall data. These interpolations are referred to as *pchips*, or Piecewise Cubic Hermite Interpolating Polynomials (Moler, 2012). A particular quality of pchips is that they act locally, producing less-exaggerated curves than conventional cubic splines on a

global scale (Figure 1). This also results in curves that look less like a polynomial, which is important considering shelf and floor regions tend to be rather flat.

Finally, it's possible to take an existing equation that represents an expected profile and perform a *non-linear fit*. The three areas of interpolation fit one characteristic equation pretty well, commonly referred to as a *logistic* equation, with two plateaus and a distinguished slope (Figure 2). The standard form of a logistic equation is of the form

$$\frac{dy}{dx} = r * y(1 - \frac{y}{k})$$

which is a first-order differential equation where r is representative of a growth rate of increasing position towards the shoreline x, and k is the asymptote representing sea level. This can be rewritten to solve for the depth, y

$$y = \frac{ky_0}{y_0 + (k - y_0)e^{-rx}}$$

which incorporates the depth of the basin floor y_0 (Powell, 2002). By altering the value of r, controlling slope steepness, these processes can be illustrated by altering slope steepness (Figure 2). Key processes that control the profile shape come from wave action and sediment deposition from rivers, acting to steepen or flatten the slope respectively.

Now, each of these methods have a set of operating parameters that they need to function. One thing in particular would be the resolution of the data provided, or how many data points are given to create a profile. Sampling done by a transducer is traditionally done in Herz, using a time domain. However, a more visually intuitive means of regulating resolution would be to use distance. For this project, data was acquired as a NMEA (National Marine Electronics Association) string containing a depth with both a timestamp and a position. The NMEA data could then undergo *downsampling*, or removal of every few data points to lower the resolution of supplied data to the model. Downsampling might be beneficial if there is noise in a dataset or the model can't capture trends at a high resolution.

Methods

This project consisted of three unique aspects: Sensor design, data acquisition, and model iteration. Each requires a unique schedule and methodology, where sensor design focused on cost-effectiveness and accuracy, data acquisition to survey and correct data for known errors, and modeling to derive a function that could describe the principle forces acting on coastal bathymetry.

In developing a sampling system for the field, a number of considerations and restraints would need to be made. The system would need to be low cost due to student budget insufficiencies. A portable, mountable system would be very advantageous when considering the range of operating vessels available. Having components readily available would also be

beneficial for testing and deployment as soon as possible. The final product would feature many reused parts (Figure 12).

The use of field data required the development of a system capable of collecting depth readings along an ideally straight transect line, which could then be broken up into profiles from shallow to deep. A number of components were already available from the Ocean Technology Center and Dave Thorson, namely the Garmin fish finder, mounting equipment, and antenna. In order to make this system deployable on a small boat like the *Welander*, it would need to act on its own power. Two 12v 14Ah batteries were used in series to supply a constant 24-26v supply to the Garmin and associated components. A transducer was available with the pre-existing Garmin fishfinder, however it has a broken mount and was replaced by a dual frequency (77 and 200kHz) Garmin transducer for a secure connection on the mounting rod.

The design was tested in the field at Mukilteo and Duckabush on the 28th and 31st of January, respectively. At Mukilteo, the goal of achieving straight transect lines was not accomplished, but was resolved at the Duckabush by providing coordinates to the *Welander* operator. In addition, data gaps appeared in the Mukilteo single-beam dataset, which were resolved by manually adjusting the Garmin 440s gain to <50% for depths <20m, and increasing the gain to >50% for depths >20m (Figure 3). An Arduino Uno DEV-1395 and LSM303DLHC IMU module were used at Duckabush to collect measurements of pitch and roll, considering the impacts they may have by moving the transducer affixed to the *Welander*. However, little to no correlations were found between these parameters as the variance for many transects was less than 11%, greatest at the deep end of 5th transect (Figure 4).

The system was then tested for accuracy in the UW Oceanography Testing Tank on the 23rd of February (Table 1). Key results from the test tank were that the Garmin 440s and transducer will return an offset of ~0.14 - 0.2m from a lead line measurement, a width in the cone of approximately 18-25 degrees, the transducer can collect readings through materials such as plastic and wood, and that the system will log the greatest depth within the cone. These accuracy observations are in line with what a fish finder would be expected to perform, which would be to collect readings through small debris to return the deepest reading.

Once single-beam data was acquired from Mukilteo and Duckabush, the active line containing positional and depth data was saved to the 4GB Emtec SD card on the Garmin 440s. This NMEA string was able to be imported to Garmin Homeport software, and then be exported as a GPX Json. This GPX data was then able to be parsed using Python into a simple CSV format, capable of being read by Microsoft Excel or MATLAB. Terrestrial and deeper marine values were acquired from Global Multi-Resolution Topography (GMRT) through Columbia University's GeoMapApp (Ryan et al. 2009). These values use land elevation from the Shuttle Radar Topography Mission (SRTM) of 2000 and many different data sources for high-resolution (~100m grid spacing) marine bathymetry. These data often encounter errors near the shallows where land topography and multibeam overlap (Figure 4), so values below zero were thrown out and supplemented with single-beam field data for a true transect.

The GMRT and field bathymetry were then imported into MATLAB for model development.

Results

In the field, the design functioned as intended with a few minor inconveniences. The depth was sampled and logged during each outing at Mukilteo and Duckabush, though they present regions where readings could not be ascertained (Figure 3). Usage of the IMU at the Duckabush delta showed roll and pitch values that did not often exceed ±10° (Figure 11). Though the conditions were unfavorable, pitch and roll are unlikely to account for the variation of that transect.

Upon implementing the three unique modeling methods, each performed uniquely to the others. First and foremost, the logistic *nlinfit* regression was scrapped since the populated matrix of interpolated elevation values was nearly blank for all datasets provided (Figure 13). The following results will omit this method, for some discussion in the conclusion.

All of the original, highest-resolution sampling plots in Figure 6 were suggesting profiles that were too deep, prompting the use of downsampling to better capture the general trends. Downsampling was divided into two new datasets. “High”, representing every 5 points or higher resolution, and “low” of every 20 data points.

The linear interpolations performed the best of the original and "high" downsampled data sets for the Duckabush only. Original (not downsampled) plots along transect 6 of the Duckabush data showed a difference in 7,326 m^2 between the raw data and the linear interpolation. This difference dropped further for the "high" downsampled interpolation, with a difference of 4,690 m^2. Though this dropped yet further to 3,014 m^2 for the "low" downsampled plot, Pchip performed better. The linear interpolations, however, were very poor for the Mukilteo dataset with a difference that exceeded 14,000 m^2 for all three downsampled sets. The linear interpolation performed better than the other methods when the profile was flattest with a high amount of sample points.

The spline interpolations were unique for a number of reasons. Firstly, they were considerably less characteristic than either of the other two methods when applied to the Duckabush, 3.4 to 6.0 times the overestimated volume the linear acquired and 2.5 to 59.6 times the overestimated volume that Pchip acquired. Lowering the sampling rate did not have results, dropping from 25,217 m^2 for the original data to 18,068 m^2 when downsampled to every 20th data point. However, it was the best integrated-method all-around for the Mukilteo set, with 9,540 m^2 difference when not downsampled. For Mukilteo, it actually estimated a shallower overall profile when downsampled, something neither the linear nor Pchip demonstrated. That being said, all of the spline plots rose out of the water again (exceeded $y = 0$). These interpolations could all be deemed as uncharacteristic, especially for the "low" downsampled set which rose to be 60 m out of the water.

Finally, Pchip did very well for all interpolations. At Duckabush, it was consistently second best in terms of volume difference from raw data except for "low" downsampling, where it only had 303 m^2 of unaccounted space. It showed a similar curvature to the spline without as exaggerated of a profile, and was able to come within ~3000-4000 m^2 of the linear fit for the non- and "high" downsampled sets. It performed perhaps the best of the three tested methods at Mukilteo, where it not only demonstrated some of the curvature of the raw data without breaking back out of the water, but also had a smaller difference in area from the raw data. It was best when given the "high" downsampled set, where it left 7,360 m^2 unaccounted for.

Conclusions

From the different interpolation methods used, different models performed best for the relatively linear Duckabush delta and the steep Mukilteo slope. For the delta terrain of Duckabush, a linear fit was most appropriate in preserving both the curvature and area differential from the observed profile. This was apparent for all data sets but the “low” run where only every 20th data point was used. In that case, pchip demonstrated a better curve which flattened and then dropped, whereas the linear fit was point-to-point and very sharp. It could be that in providing more data points, non-linear methods of interpolating the transect profiles struggle with the White Zone because they function less locally than the linear fit. Ergo, since the White Zone for the delta is rather constant overall, the linear interpolation would fit best. This is further evidenced by the surface-area drops between downsampling; the linear fit dropped by less than 4,000 m^2 from no downsampling to every 20th point. The spline and pchip both modeled the data closer when downsampled further, and showed changes of ~7,000 and ~10,000m^2 drops in surface area difference respectively.

The linear interpolation, however, was the worst choice for the Mukilteo profile. The linear method cut out all of the shelf-width area, and therefore always had a profile at a lower elevation than in the observed data. The best for Mukilteo was pchip, which acted similarly to the spline without having a profile that came back out of the water. It performs best when downsampled to every 5th point, but it still better than the others for other amounts of data points provided.

The logistic method did not work, unfortunately. Using initialization parameters of [25, -81, 4] for the land maximum, sea minimum, and slope value *r*, respectively, the resulting elevation matrix contained sparse values to plot for a number of variations on the logistic equation. This may have been due to restrictions of x values provided, improper definition of the functions within the function handles, a poor initial parameters, or that *nlinfit* didn’t cope very well with the gap between land and sea.

Regardless, the use of an interpolation schema for multibeam transects could potentially plot the shallows in real-time on a shipboard vessel, thereby providing insight to the bathymetry and further highlighting anomalous trends in the proposed model. Every model clearly contains errors due to simplifications and assumptions. For example, a resedimentation event such as a scarp left over from a submarine landslide, would not be considered an acting process, thereby generating an abnormal profile which would be hard to distinguish on land topography and

multibeam swath alone. Another assumption was that the model would not be used during or after a storm, when waves can move lots of sediment and form bars along the coastline. By identifying anomalies, coastal discoveries or survey locations can be made available to the greater scientific body as well as provide insight to coastal regions that have not been mapped before. To understand a natural process, one needs to first identify an event and where it takes place.

In conclusion, a linear fit of data points works best for shallow-water deltas at high amounts of data provided. This might be impractical, because using every 20th data point may be less intensive on the acting computer, so a pchip would actually be best for the shallow waters. If that is true, then pchip would also be used for steep bluff-backed beaches, as it consistently maintains the curvature and has the overall surface-area volume missing. Splines should be avoided, as they tend to act a little too dynamically with wild changes in slope elevation and thereby put area calculations into question.

References

Hampson, Robert, et al. “A Low-Cost Hydrographic Kayak Surveying System.” Journal of Coastal Research, vol. 27, 2011, pp. 600–603., doi:10.2112/jcoastres-d-09-00108.1.

Moler, Cleve. “Splines and Pchips” MathWorks Blogs. 2012. The Mathworks Inc. https://blogs.mathworks.com/cleve/2012/07/16/splines-and-pchips/

National Oceanic and Atmospheric Administration. “How Much of the Ocean Have We Explored?” NOAA's National Ocean Service, 1 Jan. 2009, oceanservice.noaa.gov/facts/exploration.html.

Neumann, Barbara, et al. “Future Coastal Population Growth and Exposure to Sea-Level Rise and Coastal Flooding - A Global Assessment.” Plos One, vol. 10, no. 3, 2015, doi:10.1371/journal.pone.0118571.

Powell, James. “Analytic Solution of the Logistic Equation.” Utah State University Math & Statistics Department, 2002, math.usu.edu/~powell/biomath/mlab3-02/node3.html.

Ryan, William B. F., et al. “Global Multi-Resolution Topography Synthesis.” Geochemistry, Geophysics, Geosystems, vol. 10, no. 3, 2009, doi:10.1029/2008gc002332.

Said, N. M., et al. “Satellite-Derived Bathymetry: Accuracy Assessment On Depths Derivation Algorithm For Shallow Water Area.” ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-4/W5, 2017, pp. 159–164., doi:10.5194/isprs-archives-xlii-4-w5-159-2017.

Shipman, H. 2008. *A Geomorphic Classification of Puget Sound Nearshore Landforms*. Puget Sound Nearshore Partnership Report No. 2008-01. Published by Seattle District, U.S. Army Corps of Engineers, Seattle, Washington.

Weisstein, Eric W. "Cubic Spline." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/CubicSpline.html

Appendix

Figures

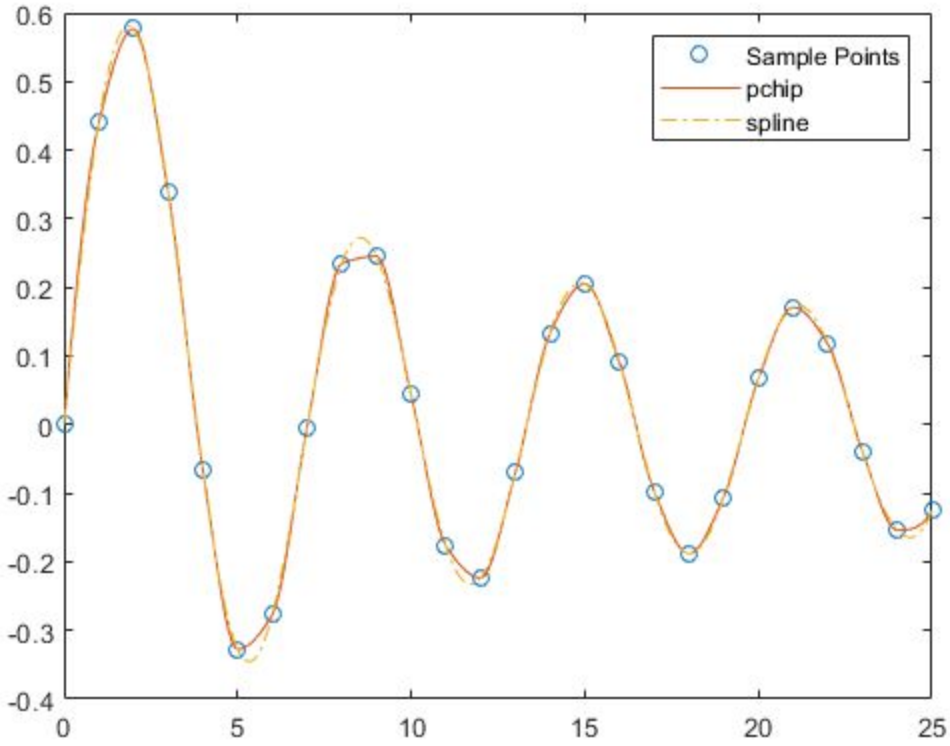

Figure 1. This figure demonstrates the difference in functionality between the pchip and conventional cubic splines for a simple Bessel curve. For demonstration, no units are provided on the x or y axes. Notice around $x = 5$ and $x = 8$ how the pchip acts locally without exhibiting the polynomial behavior of the spline. This figure was generated in MATLAB using *pchipExample.m* in the appendix.

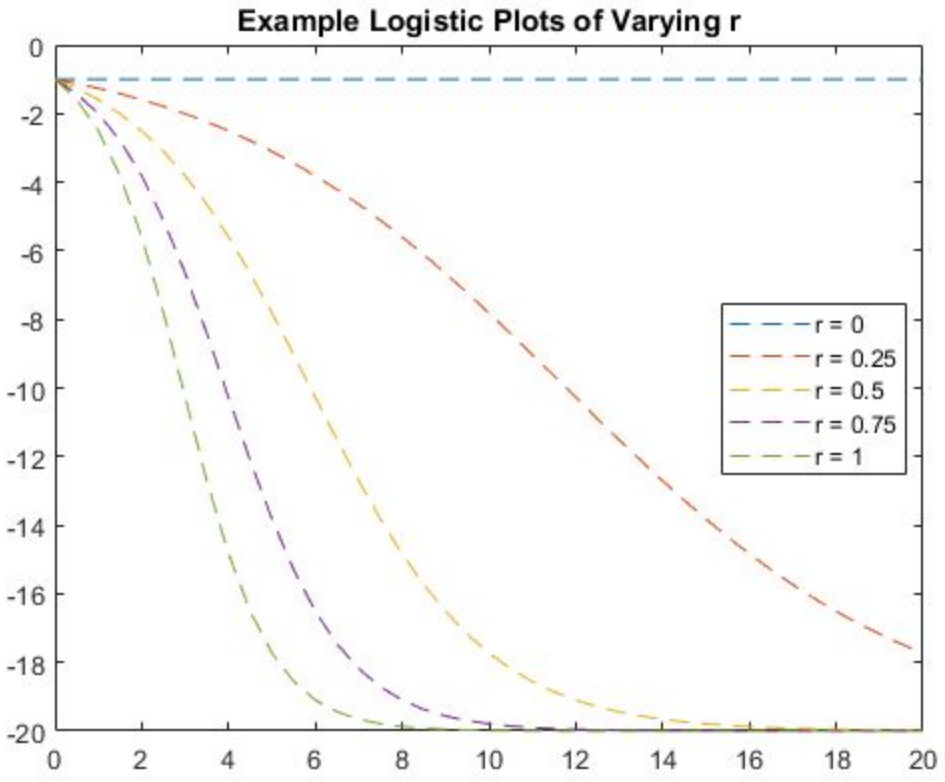

Figure 2. An example of the logistic equation, using variable slopes *r*. Initial elevation here is -1, with a maximum at -20. Having a higher value of *r* will result in a much steeper profile, as expected to be at a bluff like Mukilteo. This plot was generated by *logdemo.m*, located in the appendix.

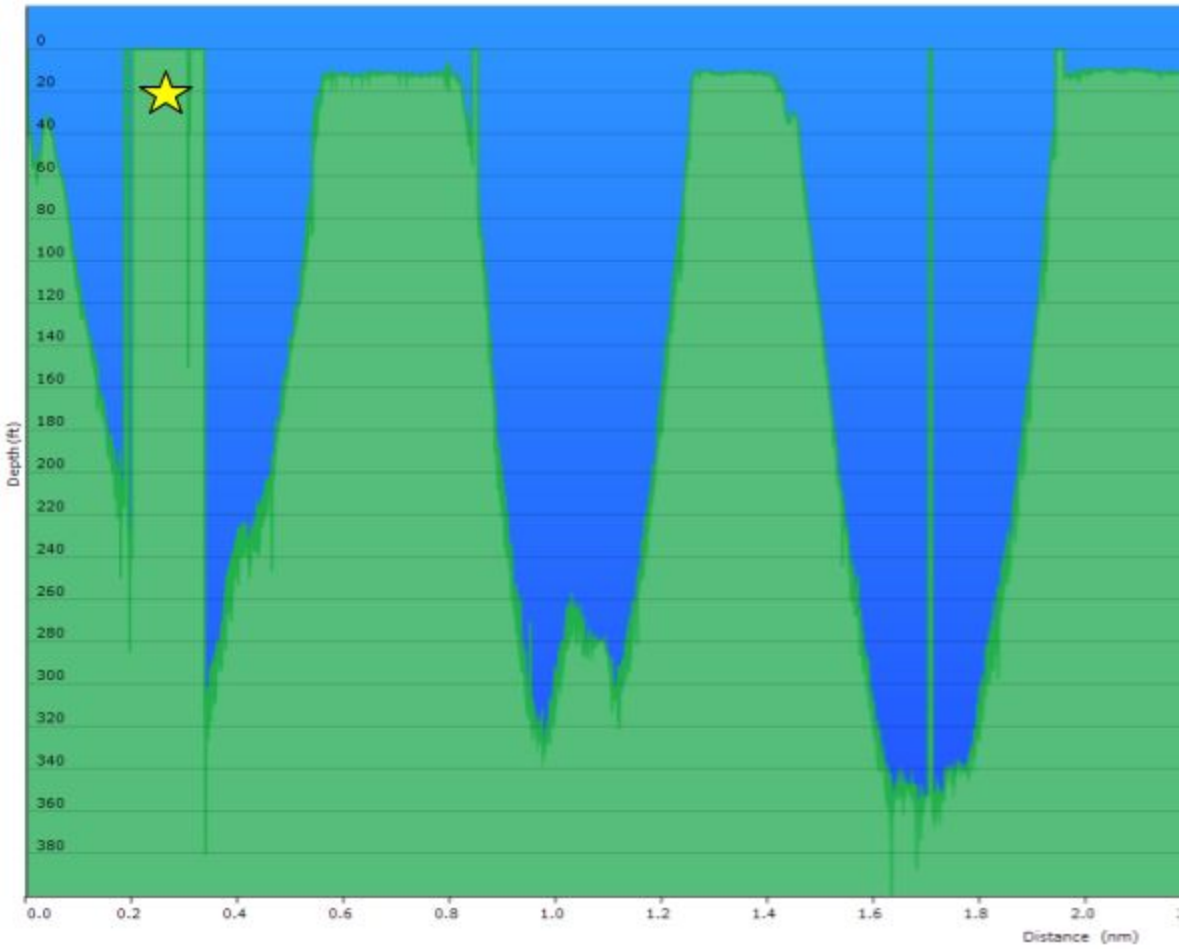

Figure 3. The transect plot of the first 6 transects at the Duckabush delta, plotted within Garmin Homeport. The vertical axis is the depth in feet, with a distance along the transect on the horizontal axis in nautical miles. Variance is greatest at the third trough (approx. 1.63 nautical miles on the x-axis) of approx. 40ft at a depth of around 350ft, or 11.4%. The first trough is not considered due to the large anomaly in the gain just before (starred), which could have influenced transducer readings.

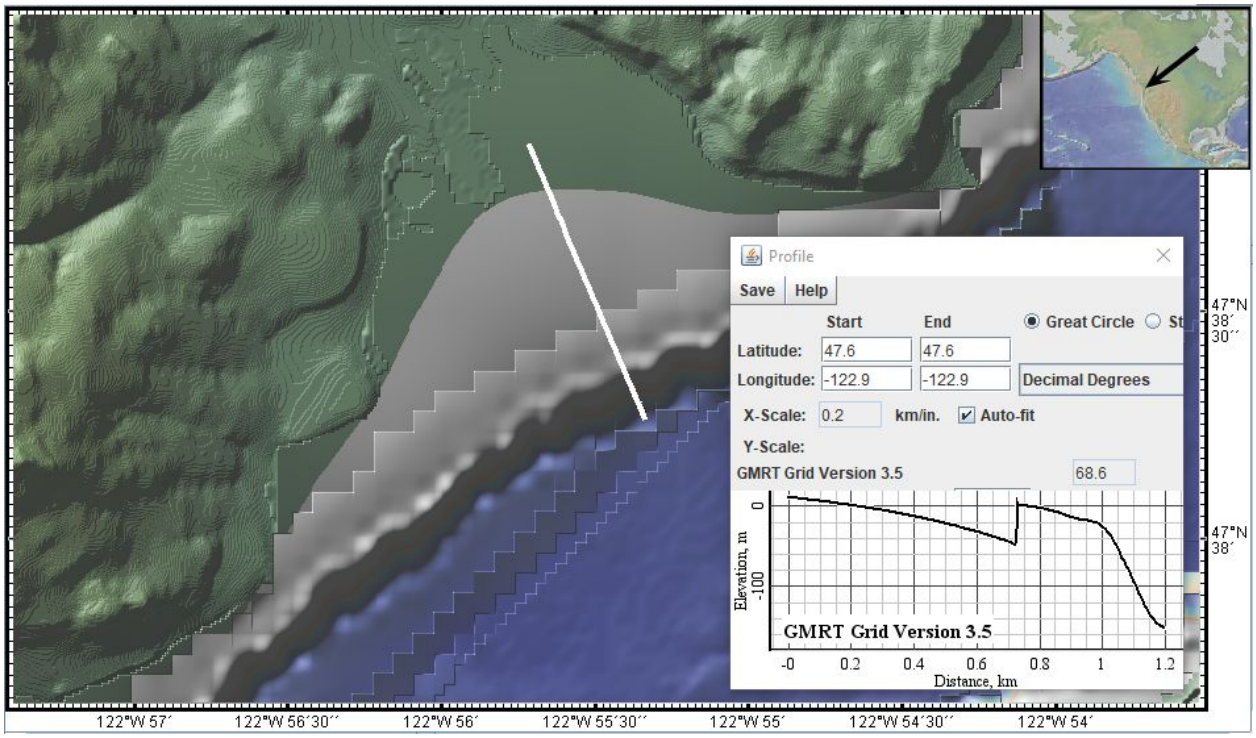

Figure 4. A bathymetry transect of the Duckabush delta in Hood Canal, WA. The depth profile at the right demonstrates a peak in the data where the GMRT Shuttle and ship multibeam datasets meet. Values in this shallow region were thrown out and replaced with acquired field data for a more coherent cross section.

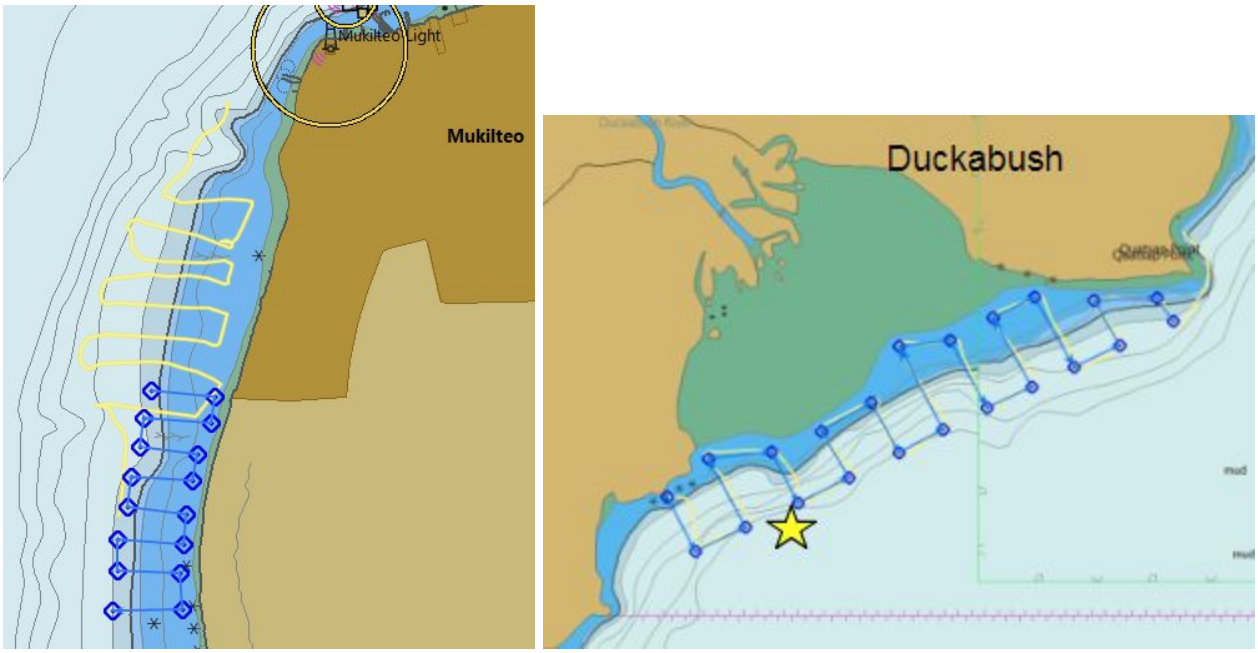

Figure 5. Sample areas of transect locations. Left is Mukilteo, where the sample region (white line) was considerably off from the ideal locations to investigate a landslide (blue dotted line). The Duckabush transect at right was much better in this regard. Starred on the Duckabush transect

is line 3, where the IMU was turned on. These are screengrabs from OpenCPN navigational software.

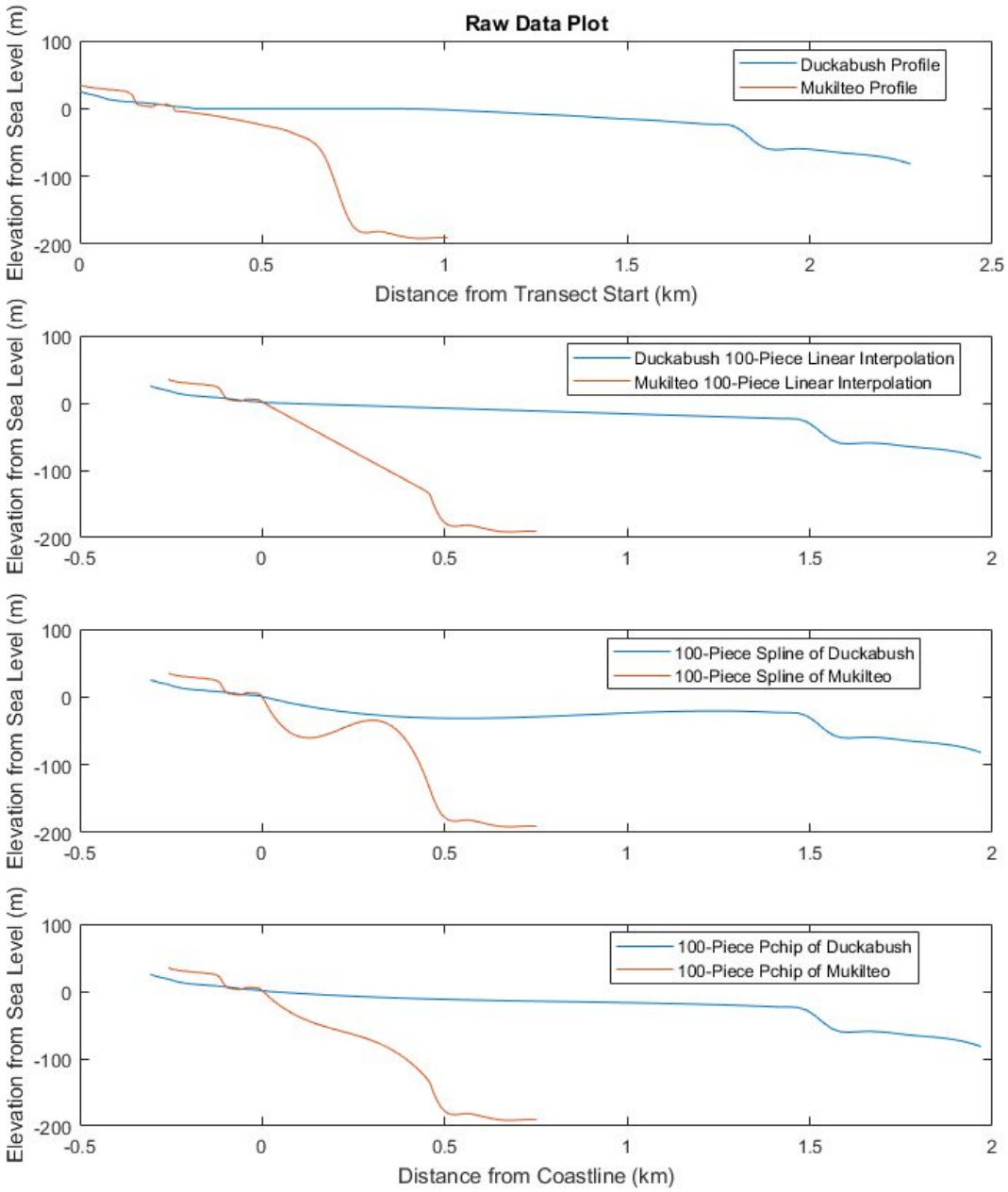

Figure 6. Plots of the three applied interpolation methods. From top to bottom; raw data, linear, spline, pchip. The Duckabush is always plotted in blue, whereas the Mukilteo is always plotted in red. Profile elevations above sea level are in meters, and horizontal scale is in kilometers. Note that the x-axis for the observed (raw) data is a distance from the beginning of the transect, rather than a distance from the coastline which all of the other plots utilize. These figures were generated in code 1.

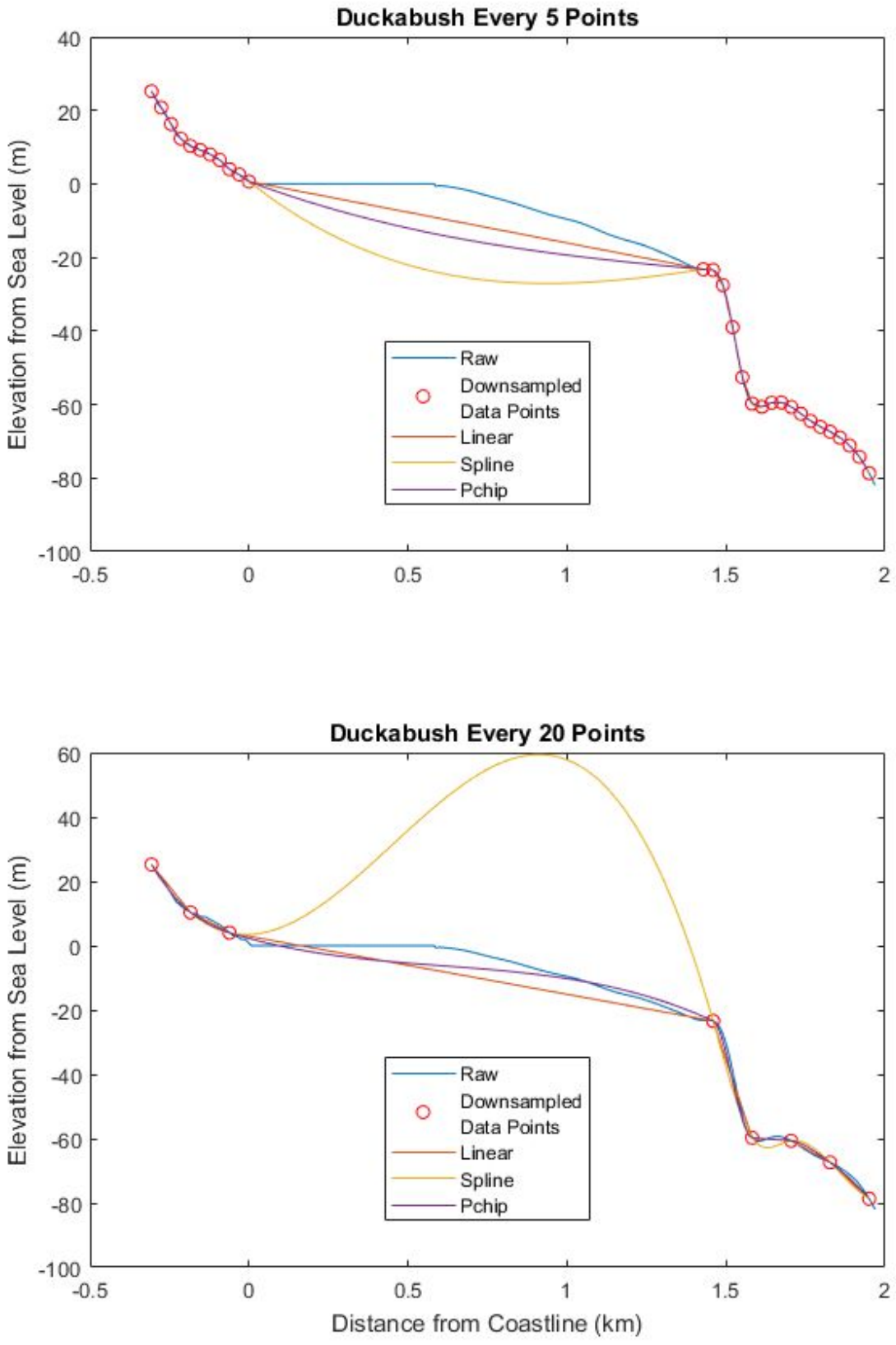

Figure 7. Plots of observed data (blue) and the linear (red), spline (yellow), and pchip (purple) interpolation methods. Profile elevations are in meters, and horizontal scale is in kilometers.

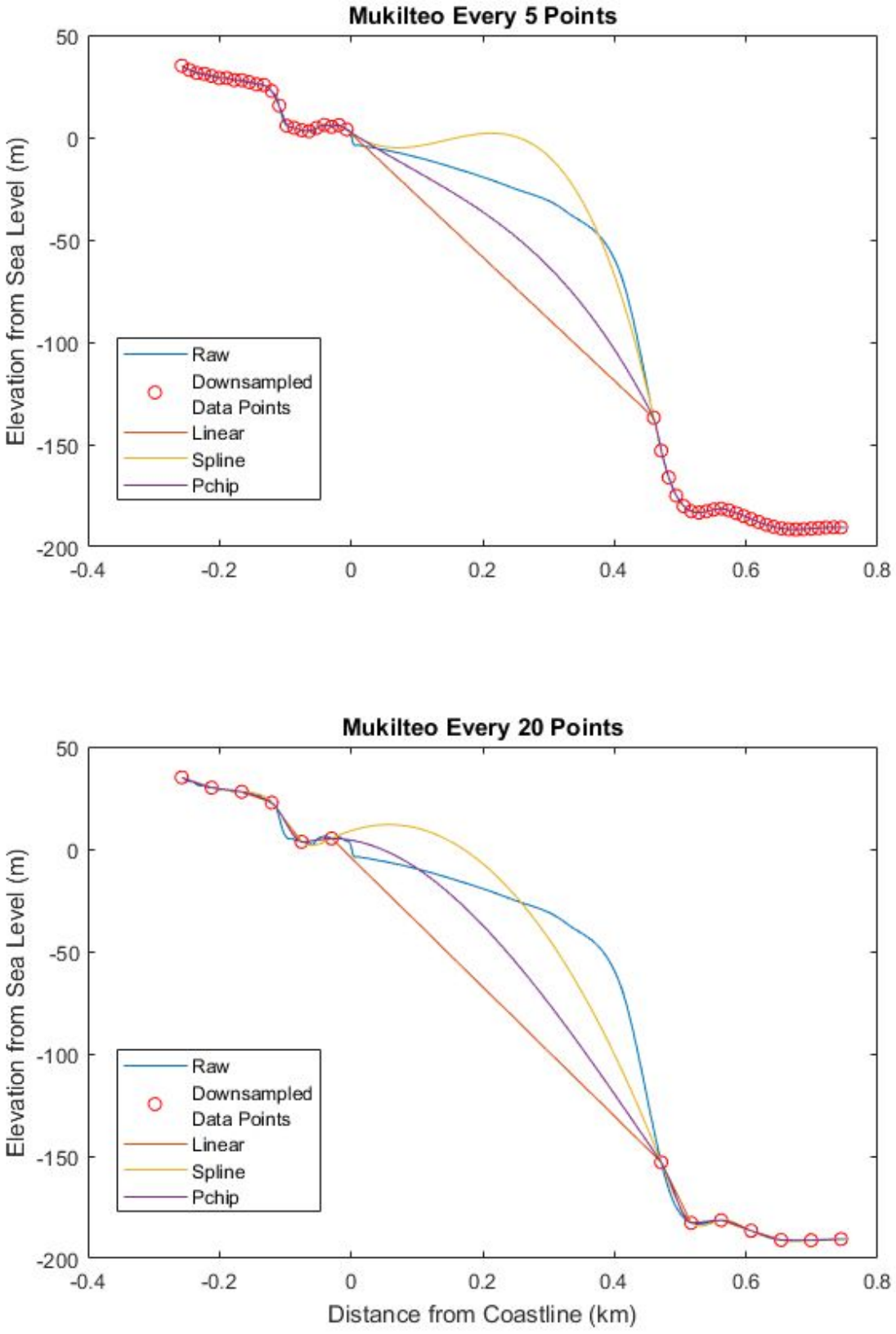

Figure 8. Plots of observed data (blue) and the linear (red), spline (yellow), and pchip (purple) interpolation methods. Profile elevations are in meters, and horizontal scale is in kilometers.

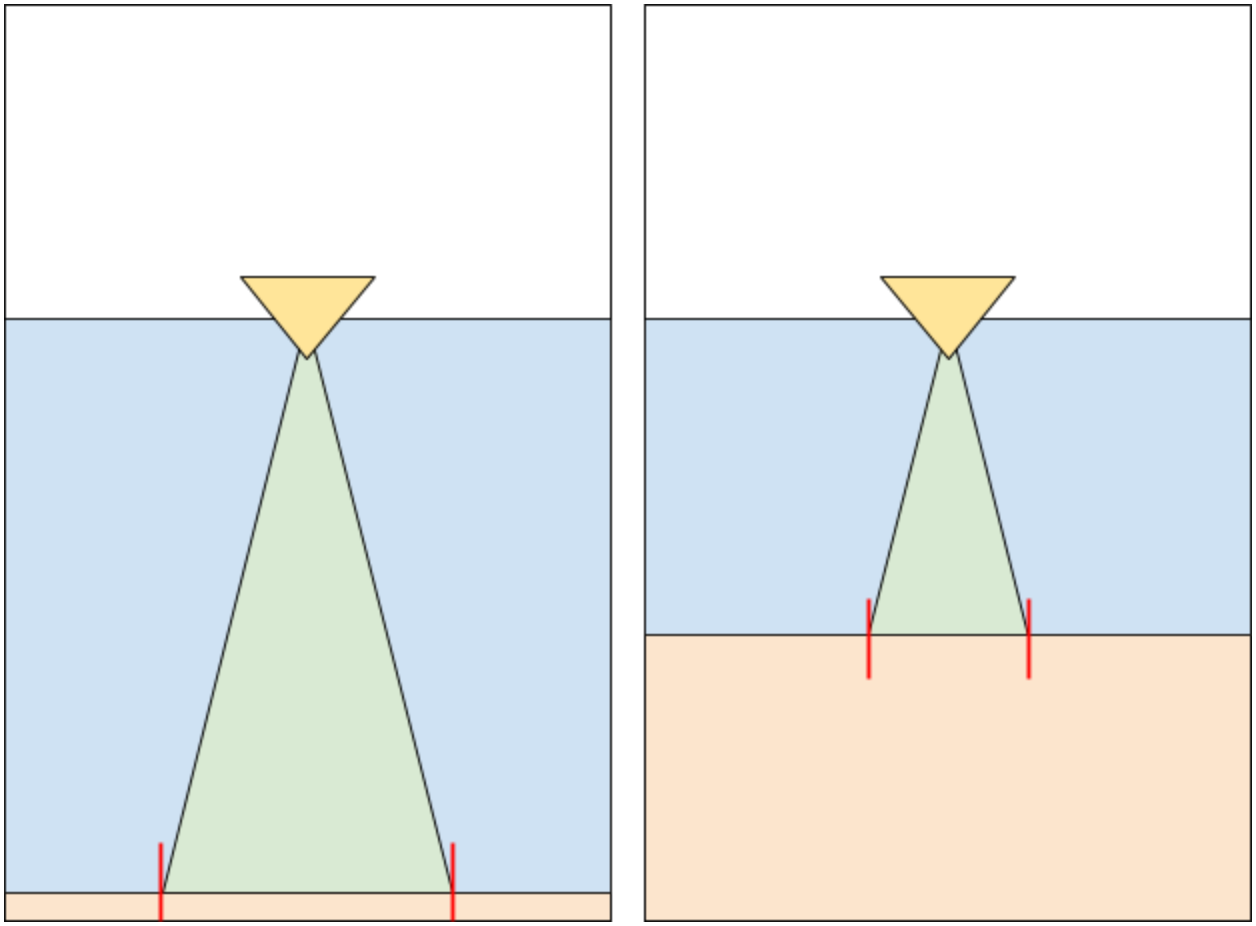

Figure 9. An illustration of multibeam swath. As the depth shoals, the width of the swath narrows. With a narrower swath, the vessel will therefore need to make more trips back and forth to cover the same area.

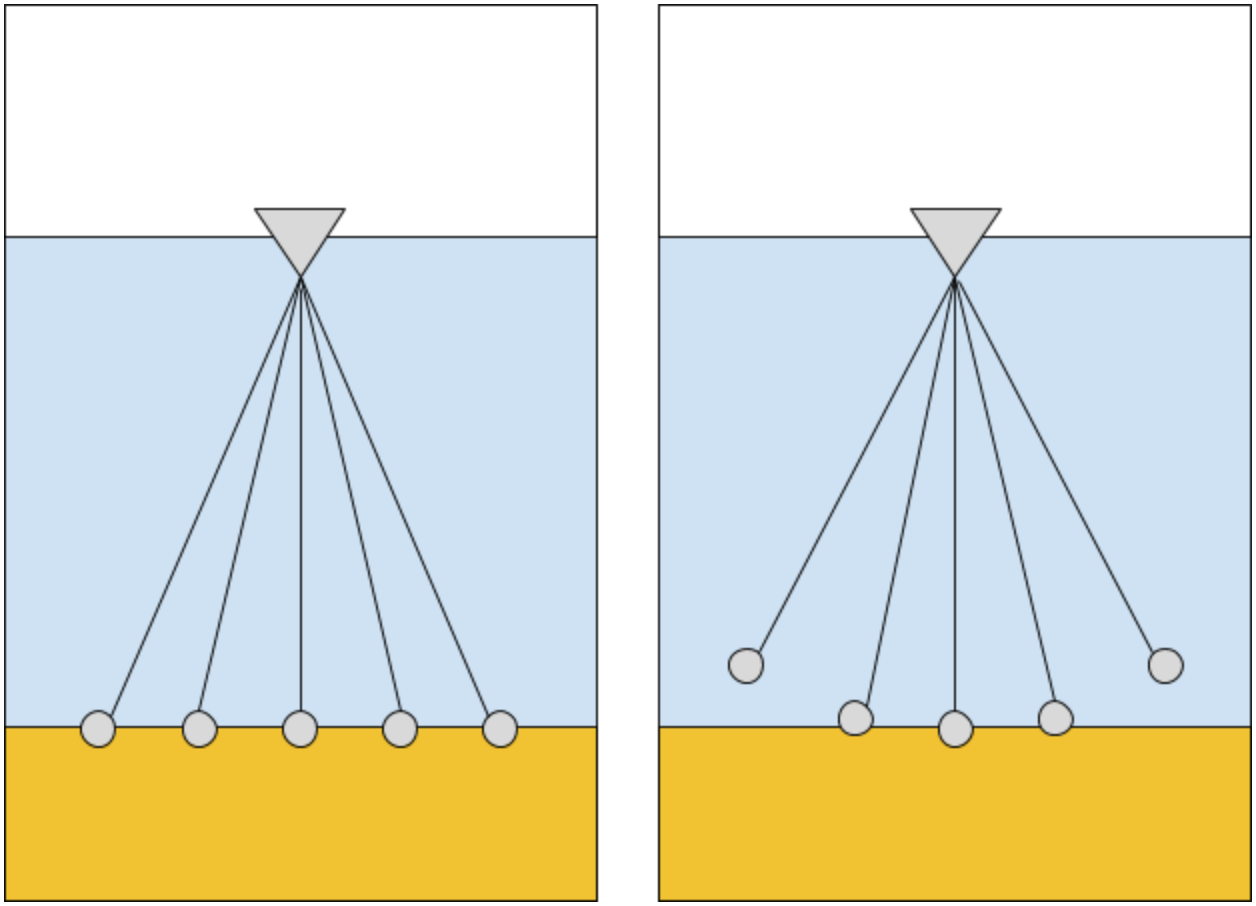

Figure 10. An illustration of multibeam "smiling". In this case, the vessel has transitioned to a lower density between the left and right panels. The outer beams are most influenced, as they must cover the greatest distance from the ship to the seabed. Therefore a small change in a SVP (such as approaching a river) can have very drastic influences on accuracy of an estuary, as the outer beams will go closest to shore.

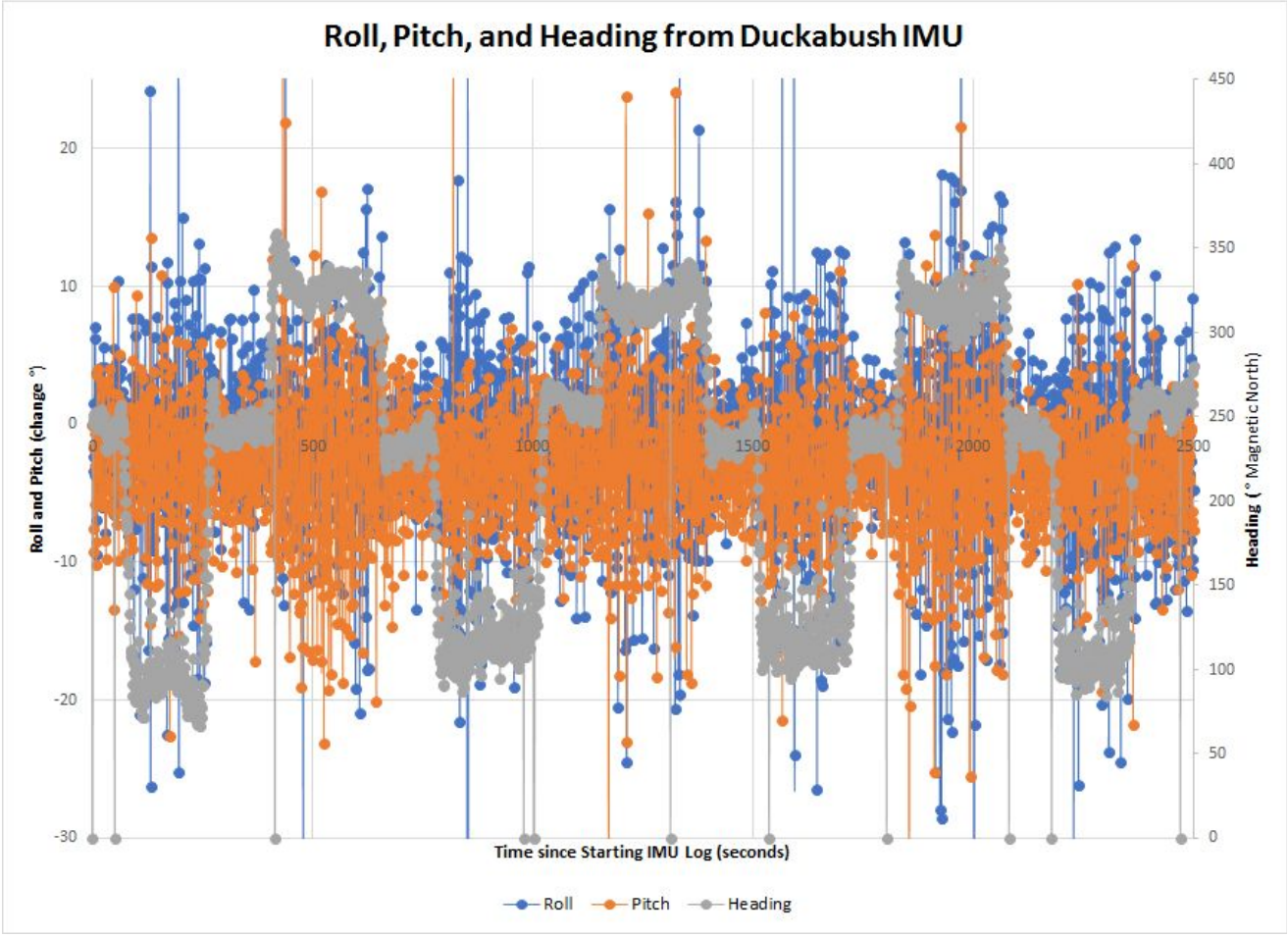

Figure 11. The IMU data from Duckabush deployment. The heading is shown in grey, pitch in orange, and roll in blue. Data has been sampled at a frequency of 1 Hz for ~2600 seconds, beginning mid-way through the third line perpendicular to the shore (starred in Figure 5). Pitch and roll infrequently exceeded ±10°, which though would move the transducer up and down slightly in the water, and is unlikely to account for the variance in Figure 3.

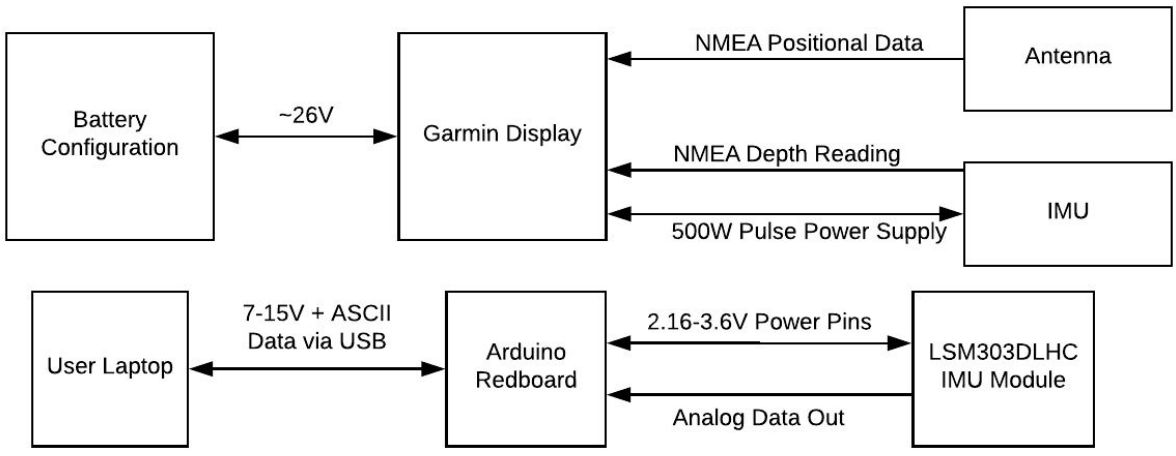

Figure 12. A simplified schematic of the components used in the field test.

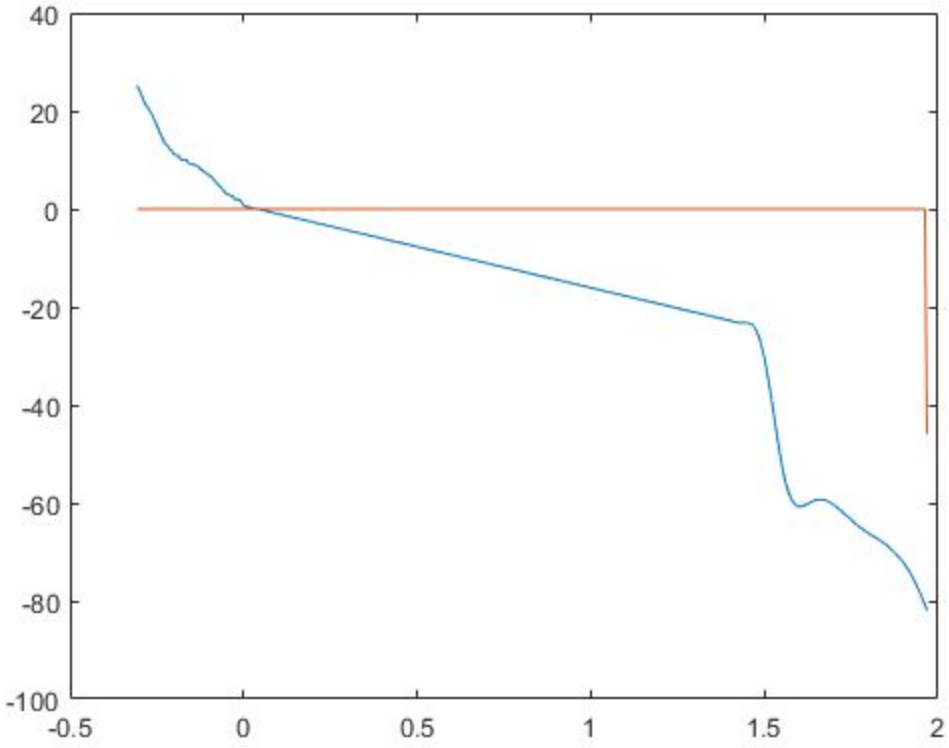

Figure 13. A rough test of MATLAB's *nlinfit* logistic for the Duckabush White Zone (interpolation in red). Values were not yet provided on the axes, but the vertical was elevation above sea level in meters, and the horizontal was in kilometers from the coastline. This method was scrapped when matrices of interpolations did not contain sufficient data points.

Tables

Correction/Testing Parameter	Finding	Notes
GPS Coordinate Provision	Much straighter transect lines	Conditions were poor on both days
Adjusted Garmin Gain	Fewer discontinuities	
Pre-Duckabush Lead Line	Offset of 0.2 m between lead line and transducer reading	
Pitch and roll overlay	Little to no influence	Conditions were poor on both days
Object Accuracy	Not found, not a reading directly underneath	Object of dimensions 15 x 15 x 38 cm
Material Return Test	Transducer blasted through plastic and wood	Plastic, Wood, and Aluminum sheet materials tested
Depth testing	Offset of 0.3 m between	Object of 78.74 x 116.84 x 0.47 cm, aluminum sheet
Wall Proximity Test	An average value is not taken from below the transducer	
Sensor Rotation Test	The greatest distance was returned Cone angle of 18 - 25°	

Table 1. A sample table of the testing performed on the Garmin 441s in both accuracy testing, and configuring between transect data collection. The all tests following and including the object accuracy test were performed on the 23rd of February.

	Area at Duckabush (m^2)			Area at Mukilteo (m^2)		
	Not Downsampled	High Sample (Every 5 points)	Low Sample (Every 20 points)	Not Downsampled	High Sample (Every 5 points)	Low Sample (Every 20 points)
Obs.	94,200			98,370		
Linear	101,526	98,890	97,210	114,577	113,190	117,260

Spline	119,417	112,760	112,270	107,910	91,940	112,270
Pchip	104,357	103,070	94,500	112,975	103,070	107,540

Table 2: This table reports the integrated areas of water in m^2 between the profile and a flat surface boundary at $y = 0$. The observed values are returned in the first row, which did not undergo any downsampling. Values were ascertained via trapezoidal integration as demonstrated in code 1 and 2, stored in the vectors *areas* in code 1, and *AreasDuck* and *AreasMuk* in code 2.

	Δ Area from Observed (Duckabush, m^2)			Δ Area from Observed (Mukilteo, m^2)		
	Not Downsampled	High Sample (Every 5 points)	Low Sample (Every 20 points)	Not Downsampled	High Sample (Every 5 points)	Low Sample (Every 20 points)
Linear	7,326	4,690	3,014	16,207	14,828	18,892
Spline	25,217	18,564	18,068	9,540	-6,430	-1,208
Pchip	10,157	8,869	303	14,605	7,360	9,170

Table 3. The differences in area of the two sample sites in m^2. The three applied methods exhibit different amounts of over- or underestimating of the water area between the surface and the bathymetry profile. Negative values, therefore, suggest that the interpolated surface as a whole was shallower than the observed data profile.

Code

Code 1: plottingcode.m

```
%Aaron Mau
%Ocean Thesis Plotting Code
%University of Washington School of Oceanography

close all; clc;

%Importing datasets into easy-to-type variables for quick plotting.
%Duckabush data, in this case transect 6 (duckline6).
x = table2array(duckline6(:,1));
y = table2array(duckline6(:,2));
%Mukilteo, transect 3.  All transects are from the origin, so this is the
%third from the northern-most section.  Transects are selected by
%straightness, because (I think) a straight profile would be easier to
%interpret than a bent one.
z = table2array(mukline3(:,1));
a = table2array(mukline3(:,2));

ylabelText = "Elevation from Sea Level (m)";
xlabelText = "Distance from Coastline (km)";

%Correcting for satellite data error (demonstrate this with attached
%GeoMapApp screenshot).  It seems like the white zone is poorly sampled
%either via satellite or multibeam swath, so I'm defining this to be a
%region of ~0 elevation.  Tidal influence is known to make this region
%highly variable, so this assumption may be justified here.
for i=52:146
    y(i)=0;
end

%Find sea level index, Y based on elevation from duckabush and A based on
%elevation from mukilteo.  The importance becomes more apparent in the next
%step.
duckCoast = find(y > 0 & y < 5);
duckCoast = duckCoast(end);
mukCoast = find(a > 0 & a < 5);
mukCoast = mukCoast(end);

%Returns the value at the coastline.  Subtract from all other values to get
%distance from the coastline.  This is useful because it can make a more
%meaningful diagram than "Distance from transect origin".
duckCoastDist = x(duckCoast);
mukCoastDist = z(mukCoast);

%Break up the datasets into land and deepwater sections.  This step could
%probably be skipped, but early on this felt like a logical step to make.
ducklandY = y(1:duckCoast);
ducklandX = x(1:duckCoast) - duckCoastDist;
duckdeepY = y(280:end);
duckdeepX = x(280:end) - duckCoastDist;

muklandY = a(1:mukCoast);
muklandX = z(1:mukCoast) - mukCoastDist;
mukdeepY = a(315:end);
mukdeepX = z(315:end) - mukCoastDist;

%Completed datasets, with the holes.  Concatinating the land (...landX/Y)
%with the deep (...deepX/Y).  These are the basis for fitting the white
```

```
%zone.
ducky = cat(1, ducklandY, duckdeepY);
duckx = cat(1, ducklandX, duckdeepX);
muky = cat(1, muklandY, mukdeepY);
mukx = cat(1, muklandX, mukdeepX);

%Downsampled datasets.  These are passed on to the interpolation methods as
%a basis of sampleable data.  If every five datapoints are provided, then
%less fine-resolution characature of the datasets will be inacted on the
%interpolation, thereby representing more of the overall trend.  Every
%twenty datapoints may be too low a resolution, but could also demonstrate
%an idealized/simplified shape for more complex models like a spline or
%high-degree polynomial.
duckdownX5 = duckx(1:5:end);  %every five datapoints
duckdownX20 = duckx(1:20:end);
duckdownY5 = ducky(1:5:end);    %need y values as well for matrix size constraints
duckdownY20 = ducky(1:20:end);

mukdownX5 = mukx(1:5:end);
mukdownX20 = mukx(1:20:end);
mukdownY5 = muky(1:5:end);
mukdownY20 = muky(1:20:end);

%get the absolute area, including all land. This is all the WATER between
%the surface and the topography profile (shifted down by the maximum).
%Comparing this later could be used to assess how closely each
%interpolation comes to modeling the real data.
duckArea = abs(trapz(x, y-max(y)));
mukArea = abs(trapz(z, a-max(a)));

% figure
% plot(x-duckCoastDist, y, 'k', duckx, ducky, 'ko', z-mukCoastDist, a, 'r', mukx, muky, 'ro');
% legend('Duckabush Complete', 'Duckabush w/o Grey', 'Mukilteo Complete', 'Mukilteo w/o Grey',
'Location', 'Best')
% xlabel(xlabelText)
% ylabel(ylabelText)

%% Linear interpolation

%into 100 pieces
duckspace100 = (duckx(end)-duckx(1))/100;   %need to base it off of the range of x values
mukspace100 = (mukx(end)-mukx(1))/100;
%into 10 pieces
duckspace10 = (duckx(end)-duckx(1))/10;
mukspace10 = (mukx(end)-mukx(1))/10;

duckxReducedHigh = (duckx(1):duckspace100:duckx(end))';
duckxReducedLow = (duckx(1):duckspace10:duckx(end))';
mukxReducedHigh = (mukx(1):mukspace100:mukx(end))';
mukxReducedLow = (mukx(1):mukspace10:mukx(end))';

ducklinHigh = interp1q(duckx, ducky, duckxReducedHigh);
ducklinLow = interp1q(duckx, ducky, duckxReducedLow);
muklinHigh = interp1q(mukx, muky, mukxReducedHigh );
muklinLow = interp1q(mukx, muky, mukxReducedLow);

% figure
% subplot(1,2,1);
% plot(duckxReducedLinHigh, ducklinHigh, mukxReducedLinHigh, muklinHigh);  %linear interpolation
at high degree
% ylabel(ylabelText)
```

```
% xlabel(xlabelText)
% legend('Duckabush High Linear Interpolation', 'Mukilteo High Linear Interpolation',
'Location', 'Best')
%
% subplot(1,2,2);
% plot(duckxReducedLinLow, ducklinLow, mukxReducedLinLow, muklinLow);
% xlabel(xlabelText)
% legend('Duckabush Low Linear Interpolation', 'Mukilteo Low Linear Interpolation', 'Location',
'Best')

ducklinAreaH = abs(trapz(duckxReducedHigh, ducklinHigh - max(ducklinHigh)));  %Areas are bigger
(more water) so the bathymetry is too low
ducklinAreaL = abs(trapz(duckxReducedLow, ducklinLow - max(ducklinLow)));
muklinAreaH = abs(trapz(mukxReducedHigh, muklinHigh - max(muklinHigh)));
muklinAreaL = abs(trapz(mukxReducedLow, muklinLow - max(muklinLow)));

%% Using a spline

%High sample frequency (100 pieces)
ducksplHigh = spline(duckx, ducky, duckx(1):duckspace100:duckx(end));
muksplHigh = spline(mukx, muky, mukx(1):mukspace100:mukx(end));

%Low sample frequency (10 pieces)
ducksplLow = spline(duckx, ducky, duckx(1):duckspace10:duckx(end));
muksplLow = spline(mukx, muky, mukx(1):mukspace10:mukx(end));

% figure
% plot(duckx, ducky,'o', duckxReducedHigh, ducksplHigh); hold on;
% plot(mukx, muky, 'o', mukxReducedHigh, muksplHigh);
% legend('Duckabush Values', 'High Res Spline of Duckabush', 'Mukilteo Values', 'High Res Spline
of Mukilteo')
%
% figure
% plot(duckx, ducky,'o', duckxReducedLow, ducksplLow); hold on;
% plot(mukx, muky, 'o', mukxReducedLow, muksplLow);
% legend('Duckabush Values', 'Spline of Duckabush', 'Mukilteo Values', 'Spline of Mukilteo')

ducksplAreaH = abs(trapz(duckxReducedHigh, ducksplHigh - max(ducksplHigh)));  %Areas are bigger
(more water) so the bathymetry is too low
ducksplAreaL = abs(trapz(duckxReducedLow, ducksplLow - max(ducksplLow)));
muksplAreaH = abs(trapz(mukxReducedHigh, muksplHigh - max(muksplHigh)));
muksplAreaL = abs(trapz(mukxReducedLow, muksplLow - max(muksplLow)));

%% PCHIP

%High sample frequency (100 pieces)
duckpcpHigh = pchip(duckx, ducky, duckx(1):duckspace100:duckx(end));
mukpcpHigh = pchip(mukx, muky, mukx(1):mukspace100:mukx(end));

%Low sample frequency (10 pieces)
duckpcpLow = pchip(duckx, ducky, duckx(1):duckspace10:duckx(end));
mukpcpLow = pchip(mukx, muky, mukx(1):mukspace10:mukx(end));

% figure
% plot(duckx, ducky, 'o', duckxReducedHigh, duckpcpHigh); hold on;
% plot(mukx, muky, 'o', mukxReducedHigh, mukpcpHigh)
%
% figure
% plot(duckx, ducky, 'o', duckxReducedLow, duckpcpLow); hold on;
% plot(mukx, muky, 'o', mukxReducedLow, mukpcpLow)
```

```
duckpcpAreaH = abs(trapz(duckxReducedHigh, duckpcpHigh - max(duckpcpHigh)));  %Areas are bigger
(more water) so the bathymetry is too low
duckpcpAreaL = abs(trapz(duckxReducedLow, duckpcpLow - max(duckpcpLow)));     %Does a better job
of capturing the characteristic slope
mukpcpAreaH = abs(trapz(mukxReducedHigh, mukpcpHigh - max(mukpcpHigh)));
mukpcpAreaL = abs(trapz(mukxReducedLow, mukpcpLow - max(mukpcpLow)));

%% Proper downsampling

% % As a reference:
% % duckdownX5 = duckx(1:5:end);  %every five datapoints
% % duckdownX20 = duckx(1:20:end);
% % duckdownY5 = ducky(1:5:end);    %need y values as well for matrix size constraints
% % duckdownY20 = ducky(1:20:end);
% %
% % mukdownX5 = mukx(1:5:end);
% % mukdownX20 = mukx(1:20:end);
% % mukdownY5 = muky(1:5:end);
% % mukdownY20 = muky(1:20:end);
% duckdx5 = duckdownX5(1):0.01:duckdownX5(end);
% duckspline5 = spline(duckdownX5, duckdownY5, duckdx5);
% duckpchip5 = pchip(duckdownX5, duckdownY5, duckdx5);
%
% mukdx5 = mukdownX5(1):0.01:mukdownX5(end);
% mukspline5 = spline(mukdownX5, mukdownY5, mukdx5);
% mukpchip5 = pchip(mukdownX5, mukdownY5, mukdx5);
%
% duckdx20 = duckdownX20(1):0.01:duckdownX20(end);
% duckspline20 = spline(duckdownX20, duckdownY20, duckdx20);
% duckpchip20 = pchip(duckdownX20, duckdownY20, duckdx20);
%
% mukdx20 = mukdownX20(1):0.01:mukdownX20(end);
% mukspline20 = spline(mukdownX20, mukdownY20, mukdx20);
% mukpchip20 = pchip(mukdownX20, mukdownY20, mukdx20);
%
% figure('Name','Downsampled Duckabush Interpolations')
% subplot(2,1,1)
% plot(x - duckCoastDist, y, duckdownX5, duckdownY5, 'ro', duckdownX5, duckdownY5, duckdx5,
ducksplne5, duckdx5, duckpchip5)
% legend('Raw', ['Downsampled' newline 'Data Points'], 'Linear', 'Spline', 'Pchip', 'Location',
'Best')
% title('Duckabush Every 5 Points')
% ylabel(ylabelText)
%
% subplot(2,1,2)
% plot(x - duckCoastDist, y, duckdownX20, duckdownY20, 'ro', duckdownX20, duckdownY20, duckdx20,
ducksplne20, duckdx20, duckpchip20)
% legend('Raw', ['Downsampled' newline 'Data Points'], 'Linear', 'Spline', 'Pchip', 'Location',
'Best')
% title('Duckabush Every 20 Points')
% ylabel(ylabelText)
% xlabel(xlabelText)
%
% figure('Name', 'Downsampled Mukilteo Interpolations')
%
% subplot(2,1,1)
% plot(z - mukCoastDist, a, mukdownX5, mukdownY5, 'ro', mukdownX5, mukdownY5, mukdx5,
mukspline5, mukdx5, mukpchip5)
% legend('Raw', ['Downsampled' newline 'Data Points'], 'Linear', 'Spline', 'Pchip', 'Location',
'Best')
% title('Mukilteo Every 5 Points')
```

```
%
% subplot(2,1,2)
% plot(z - mukCoastDist, a, mukdownX20, mukdownY20, 'ro', mukdownX20, mukdownY20, mukdx20,
muksplineZ0, mukdx20, mukpchip20)
% legend('Raw', ['Downsampled' newline 'Data Points'], 'Linear', 'Spline', 'Pchip', 'Location',
'Best')
% title('Mukilteo Every 20 Points')
% xlabel(xlabelText)
%
% % yi5 = interp1(duckdownX5, duckdownY5, duckdownX5(1):0.08:duckdownX5(end));
% % ducklinArea5 = abs(trapz(duckdownX5, yi5 - max(yi5)));
%
% yi5 = interp1(duckdownX5, duckdownY5, duckdx5);
% ducklinArea5 = abs(trapz(duckdx5, yi5 - max(yi5)));
% ducksplArea5 = abs(trapz(duckdx5, duckspline5 - max(duckspline5)));
% duckpcpArea5 = abs(trapz(duckdx5, duckpchip5 - max(duckpchip5)));
%
% yi20 = interp1(duckdownX20, duckdownY20, duckdx20);
% ducklinArea20 = abs(trapz(duckdx20, yi20 - max(yi20)));
% ducksplArea20 = abs(trapz(duckdx20, duckspline20 - max(duckspline20)));
% duckpcpArea20 = abs(trapz(duckdx20, duckpchip20 - max(duckpchip20)));
%
% xi5 = interp1(mukdownX5, mukdownY5, mukdx5);
% muklinArea5 = abs(trapz(mukdx5, xi5 - max(xi5)));
% muksplArea5 = abs(trapz(mukdx5, muksplineZ5 - max(muksplineZ5)));
% mukpcpArea5 = abs(trapz(mukdx5, mukpchip5 - max(mukpchip5)));
%
% xi20 = interp1(mukdownX20, mukdownY20, mukdx20);
% muklinArea20 = abs(trapz(mukdx20, xi20 - max(xi20)));
% muksplArea20 = abs(trapz(mukdx20, muksplineZ0 - max(muksplineZ0)));
% mukpcpArea20 = abs(trapz(mukdx20, mukpchip20 - max(mukpchip20)));

%% Non-linear with Logistic equation



%% Doing ALL of the plotting
%Add "mukx, muky, 'o', " to any plots for circles of the actual data

%Raw data
figure('Name', 'All Data Plots')
subplot(4,1,1);
plot(x, y, z, a);
legend('Duckabush Profile', 'Mukilteo Profile', 'Location', 'Best')
title('Raw Data Plot')
ylabel(ylabelText)
xlabel('Distance from Transect Start (km)')

%Linear plotting
subplot(4,1,2);
plot(duckxReducedHigh, ducklinHigh, mukxReducedHigh, muklinHigh);  %linear interpolation at high
degree
ylabel(ylabelText)
legend('Duckabush 100-Piece Linear Interpolation', 'Mukilteo 100-Piece Linear Interpolation',
'Location', 'Best')

% subplot(4,2,4);   %This is a bad chart, broken method
% plot(duckxReducedLow, ducklinLow, mukxReducedLow, muklinLow);
% legend('Duckabush 10-Piece Linear Interpolation', 'Mukilteo 10-Piece Linear Interpolation',
'Location', 'Best')
```

```
% subplot(3,1,2);
% plot(duckx, ducky, mukx, muky)
% legend('Duckabush Profile', 'Mukilteo Profile', 'Location', 'Best')
% title('White Zone: Missing Shallow Data')
% ylabel(ylabelText)

%Splines
subplot(4,1,3);
plot(duckxReducedHigh, ducksplHigh); hold on;
plot(mukxReducedHigh, muksplHigh);
legend('100-Piece Spline of Duckabush','100-Piece Spline of Mukilteo', 'Location', 'Best')
ylabel(ylabelText)

% subplot(4,2,6);   %This is a bad chart, broken method
% plot(duckxReducedLow, ducksplLow); hold on;
% plot(mukxReducedLow, muksplLow);
% legend('10-Piece Spline of Duckabush','10-Piece Spline of Mukilteo', 'Location', 'Best')

subplot(4,1,4);
plot(duckxReducedHigh, duckpcpHigh, mukxReducedHigh, mukpcpHigh);
legend('100-Piece Pchip of Duckabush', '100-Piece Pchip of Mukilteo', 'Location', 'Best')
xlabel(xlabelText)
ylabel(ylabelText)

% subplot(4,2,8);   %This is a bad chart, broken method
% plot(duckxReducedLow, duckpcpLow, mukxReducedLow, mukpcpLow);
% legend('10-Piece Pchip of Duckabush', '10-Piece Pchip of Mukilteo', 'Location', 'Best')
% xlabel(xlabelText)

%% Integration Comparison
%Putting all of the calculated areas into a single array for easy access
%and comparison.
areas = [
duckArea;
mukArea;
ducklinAreaH;
ducklinAreaL;
muklinAreaH;
muklinAreaL;
ducksplAreaH;
ducksplAreaL;
muksplAreaH;
muksplAreaL;
duckpcpAreaH;
duckpcpAreaL;
mukpcpAreaH;
mukpcpAreaL
];
```

Code 2: DownsamplingCode.m

```
%Aaron Mau
%Ocean Thesis Plotting Code
%University of Washington School of Oceanography

close all; clc;

%Importing datasets into easy-to-type variables for quick plotting.
%Duckabush data, in this case transect 6 (duckline6).
x = table2array(duckline6(:,1));
y = table2array(duckline6(:,2));
%Mukilteo, transect 3.  All transects are from the origin, so this is the
```

```
%third from the northern-most section.  Transects are selected by
%straightness, because (I think) a straight profile would be easier to
%interpret than a bent one.
z = table2array(mukline3(:,1));
a = table2array(mukline3(:,2));

ylabelText = "Elevation from Sea Level (m)";
xlabelText = "Distance from Coastline (km)";

%Correcting for satellite data error (demonstrate this with attached
%GeoMapApp screenshot).  It seems like the white zone is poorly sampled
%either via satellite or multibeam swath, so I'm defining this to be a
%region of ~0 elevation.  Tidal influence is known to make this region
%highly variable, so this assumption may be justified here.
for i=52:146
    y(i)=0;
end

%Find sea level index, Y based on elevation from duckabush and A based on
%elevation from mukilteo.  The importance becomes more apparent in the next
%step.
duckCoast = find(y > 0 & y < 5);
duckCoast = duckCoast(end);
mukCoast = find(a > 0 & a < 5);
mukCoast = mukCoast(end);

%Returns the value at the coastline.  Subtract from all other values to get
%distance from the coastline.  This is useful because it can make a more
%meaningful diagram than "Distance from transect origin".
duckCoastDist = x(duckCoast);
mukCoastDist = z(mukCoast);

%Break up the datasets into land and deepwater sections.  This step could
%probably be skipped, but early on this felt like a logical step to make.
ducklandY = y(1:duckCoast);
ducklandX = x(1:duckCoast) - duckCoastDist;
duckdeepY = y(280:end);
duckdeepX = x(280:end) - duckCoastDist;

muklandY = a(1:mukCoast);
muklandX = z(1:mukCoast) - mukCoastDist;
mukdeepY = a(315:end);
mukdeepX = z(315:end) - mukCoastDist;

%Completed datasets, with the holes.  Concatinating the land (...landX/Y)
%with the deep (...deepX/Y).  These are the basis for fitting the white
%zone.
ducky = cat(1, ducklandY, duckdeepY);
duckx = cat(1, ducklandX, duckdeepX);
muky = cat(1, muklandY, mukdeepY);
mukx = cat(1, muklandX, mukdeepX);

%% Proper downsampling

% As a reference:
duckdownX5 = duckx(1:5:end);  %every five datapoints
duckdownX20 = duckx(1:20:end);
duckdownY5 = ducky(1:5:end);    %need y values as well for matrix size constraints
duckdownY20 = ducky(1:20:end);

mukdownX5 = mukx(1:5:end);
```

```
mukdownX20 = mukx(1:20:end);
mukdownY5 = muky(1:5:end);
mukdownY20 = muky(1:20:end);

duckdx5 = duckdownX5(1):0.01:duckdownX5(end);
ducksplineS = spline(duckdownX5, duckdownY5, duckdx5);
duckpchip5 = pchip(duckdownX5, duckdownY5, duckdx5);

mukdx5 = mukdownX5(1):0.01:mukdownX5(end);
muksplineS = spline(mukdownX5, mukdownY5, mukdx5);
mukpchip5 = pchip(mukdownX5, mukdownY5, mukdx5);

duckdx20 = duckdownX20(1):0.01:duckdownX20(end);
ducksplineS = spline(duckdownX20, duckdownY20, duckdx20);
duckpchip20 = pchip(duckdownX20, duckdownY20, duckdx20);

mukdx20 = mukdownX20(1):0.01:mukdownX20(end);
muksplineS = spline(mukdownX20, mukdownY20, mukdx20);
mukpchip20 = pchip(mukdownX20, mukdownY20, mukdx20);

figure('Name','Downsampled Duckabush Interpolations')
subplot(2,1,1)
plot(x - duckCoastDist, y, duckdownX5, duckdownY5, 'ro', duckdownX5, duckdownY5, duckdx5,
ducksplineS, duckdx5, duckpchip5)
legend('Raw', ['Downsampled' newline 'Data Points'], 'Linear', 'Spline', 'Pchip', 'Location',
'Best')
title('Duckabush Every 5 Points')
ylabel(ylabelText)

subplot(2,1,2)
plot(x - duckCoastDist, y, duckdownX20, duckdownY20, 'ro', duckdownX20, duckdownY20, duckdx20,
ducksplineS, duckdx20, duckpchip20)
legend('Raw', ['Downsampled' newline 'Data Points'], 'Linear', 'Spline', 'Pchip', 'Location',
'Best')
title('Duckabush Every 20 Points')
ylabel(ylabelText)
xlabel(xlabelText)

figure('Name', 'Downsampled Mukilteo Interpolations')

subplot(2,1,1)
plot(z - mukCoastDist, a, mukdownX5, mukdownY5, 'ro', mukdownX5, mukdownY5, mukdx5, muksplineS,
mukdx5, mukpchip5)
legend('Raw', ['Downsampled' newline 'Data Points'], 'Linear', 'Spline', 'Pchip', 'Location',
'Best')
title('Mukilteo Every 5 Points')
ylabel(ylabelText)

subplot(2,1,2)
plot(z - mukCoastDist, a, mukdownX20, mukdownY20, 'ro', mukdownX20, mukdownY20, mukdx20,
muksplineS, mukdx20, mukpchip20)
legend('Raw', ['Downsampled' newline 'Data Points'], 'Linear', 'Spline', 'Pchip', 'Location',
'Best')
title('Mukilteo Every 20 Points')
xlabel(xlabelText)
ylabel(ylabelText)

% yi5 = interp1(duckdownX5, duckdownY5, duckdownX5(1):0.08:duckdownX5(end));
% ducklinArea5 = abs(trapz(duckdownX5, yi5 - max(yi5)));

duckArea = abs(trapz(x, y-max(y)));
```

```
mukArea = abs(trapz(z, a-max(a)));

yi5 = interp1(duckdownX5, duckdownY5, duckdx5);
ducklinArea5 = abs(trapz(duckdx5, yi5 - max(yi5)));
ducksplArea5 = abs(trapz(duckdx5, duckspline5 - max(duckspline5)));
duckpcpArea5 = abs(trapz(duckdx5, duckpchip5 - max(duckpchip5)));

yi20 = interp1(duckdownX20, duckdownY20, duckdx20);
ducklinArea20 = abs(trapz(duckdx20, yi20 - max(yi20)));
ducksplArea20 = abs(trapz(duckdx20, duckspline20 - max(duckspline20)));
duckpcpArea20 = abs(trapz(duckdx20, duckpchip20 - max(duckpchip20)));

xi5 = interp1(mukdownX5, mukdownY5, mukdx5);
muklinArea5 = abs(trapz(mukdx5, xi5 - max(xi5)));
muksplArea5 = abs(trapz(mukdx5, mukspline5 - max(mukspline5)));
mukpcpArea5 = abs(trapz(mukdx5, mukpchip5 - max(mukpchip5)));

xi20 = interp1(mukdownX20, mukdownY20, mukdx20);
muklinArea20 = abs(trapz(mukdx20, xi20 - max(xi20)));
muksplArea20 = abs(trapz(mukdx20, mukspline20 - max(mukspline20)));
mukpcpArea20 = abs(trapz(mukdx20, mukpchip20 - max(mukpchip20)));

AreasDuck = [duckArea, ducklinArea5, ducksplArea5, duckpcpArea5, ducklinArea20, ducksplArea20,
duckpcpArea20];
AreasMuk = [mukArea, muklinArea5, muksplArea5, mukpcpArea5, muklinArea20, muksplArea20,
mukpcpArea20];
%Meters by kilometers is 1000m^2, so answers are in 1000s of m^2.  To
%convert to simple m^2...
AreasDuck = AreasDuck * 1000;
AreasMuk = AreasMuk * 1000;

DeltasDuck = [];
DeltasMuk = [];
for j = 1:7
    k = AreasDuck(j) - AreasDuck(1);
    l = AreasMuk(j) - AreasMuk(1);
    DeltasDuck = [DeltasDuck, k];
    DeltasMuk = [DeltasMuk, l];
end
```

Code 3: pchipExample.m

```
% This code is available on the MathWorks documentation page for pchips.
% https://www.mathworks.com/help/matlab/ref/pchip.html?searchHighlight=
% pchip&s_tid=doc_srchtitle
x = 0:25;
y = besselj(1,x);
xq2 = 0:0.01:25;
p = pchip(x,y,xq2);
s = spline(x,y,xq2);
plot(x,y,'o',xq2,p,'-',xq2,s,'-.')
legend('Sample Points','pchip','spline')
```

Code 4: logdemo.m

```
% Logistic Demo

log_fun = @(r, x) (-20 ./ (1 + (20 - 1)*exp(-r*x)));

x = 0:0.5:20;
r1 = 0;
```

```
r2 = 0.25;
r3 = 0.5;
r4 = 0.75;
r5 = 1;

y1 = log_fun(r1, x);
y2 = log_fun(r2, x);
y3 = log_fun(r3, x);
y4 = log_fun(r4, x);
y5 = log_fun(r5, x);


figure
plot(x, y1, '--', x, y2, '--', x, y3, '--', x, y4, '--', x, y5, '--')
title('Example Logistic Plots of Varying r')
legend('r = 0', 'r = 0.25', 'r = 0.5', 'r = 0.75', 'r = 1', 'Location', 'Best')
```

Code 5: Updated Plotting Code with Logistic Experimentation

```
PlottingCodeREV2.m
Earlier this month
May 8

You edited an item
Text
PlottingCodeREV2.m
May 8

You uploaded an item
Text
PlottingCodeREV2.m
% Aaron Mau
% Oceanography
% Senior thesis: Interpolation Code REV2
% 27. May 2018

clear all; close all; clc;      %Initialize the session by closing all windows, variables, and
command window.

%% Data acquisition

% Importing from the CSV files, this was specific to a machine in the SAL
duckabush6_unprocessed =
'C:\Users\TEMP\Documents\MATLAB\drive-download-20180508T203037Z-001\duckline6.csv';
mukilteo3_unprocessed =
'C:\Users\TEMP\Documents\MATLAB\drive-download-20180508T203037Z-001\mukline3.csv';
mukilteo6_unprocessed =
'C:\Users\TEMP\Documents\MATLAB\drive-download-20180508T203037Z-001\mukline6.csv';

duck_mat_6 = csvread(duckabush6_unprocessed, 2, 2);
muk_mat_3 = csvread(mukilteo3_unprocessed, 2, 2);
muk_mat_6 = csvread(mukilteo6_unprocessed, 2, 2);

% Correct for satellite error in the Duckabush dataset
for i=52:146
    duck_mat_6(i, 2)=0;
end

% Adjust all matrices to be distance from the shoreline, not a transect
% distance
```

```
% Find the coastline
duckCoast6 = find(duck_mat_6(:,2) > 0 & duck_mat_6(:,2) < 5);
duckCoast6 = duckCoast6(end);
mukCoast3 = find(muk_mat_3(:,2) > 0 & muk_mat_3(:,2) < 5);
mukCoast3 = mukCoast3(end);

% Get the transect distance at the coastline
duckCoastDist6 = duck_mat_6(duckCoast6,1);
mukCoastDist3 = muk_mat_3(mukCoast3,1);

% Find the index of multibeam limit

% Subtract transect distance from entire matrix to get distance from
% coastline
duckNew6 = [duck_mat_6(:,1) - duckCoastDist6, duck_mat_6(:,2)];
mukNew3 = [muk_mat_3(:,1) - mukCoastDist3, muk_mat_3(:,2)];

% Create new matrix with the whitezone data
duckWhite6 = duckNew6(duckCoast6:280,:);
mukWhite3 = mukNew3(mukCoast3:315,:);

duckMain6 = cat(1,duckNew6(1:duckCoast6,:),duckNew6(280:end,:));
mukMain3 = cat(1,mukNew3(1:mukCoast3,:),mukNew3(315:end,:));

%Downsampled datasets
duckHigh6 = duckMain6(1:5:end,:);   %Every 5 points, larger matrix
duckLow6 = duckMain6(1:20:end,:);   %Every 20 points, smaller matrix (low)
mukHigh3 = mukMain3(1:5:end,:);
mukLow3 = mukMain3(1:20:end,:);

a = duckHigh6(:,1);
b = duckHigh6(:,2);
%% Fitting

% Getting small DX values from downsampled datasets
duckDX6 = duckHigh6(1,1):0.01:duckHigh6(end,1);
% duckDXLow6 = duckLow6(1,1):0.01:duckLow6(end,1); % This isn't necessary
% because the high and the low both return the same iterations since they
% start and end at the same places
mukDX3 = mukHigh3(1,1):0.01:mukHigh3(end,1);
% mukDXLow3 = mukLow3(1,1):0.01:mukLow3(end,1);

duckSplineHigh6 = spline(duckHigh6(:,1), duckHigh6(:,2), duckDX6);
duckPchipHigh6 = pchip(duckHigh6(:,1), duckHigh6(:,2), duckDX6);
duckSplineLow6 = spline(duckLow6(:,1), duckLow6(:,2), duckDX6);
duckPchipLow6 = pchip(duckLow6(:,1), duckLow6(:,2), duckDX6);

muksplineHigh3 = spline(mukHigh3(:,1), mukHigh3(:,2), mukDX3);
mukPchipHigh3 = pchip(mukHigh3(:,1), mukHigh3(:,2), mukDX3);
mukSplineLow3 = spline(mukLow3(:,1), mukLow3(:,2), mukDX3);
mukPchipLow3 = pchip(mukLow3(:,1), mukLow3(:,2), mukDX3);

% Defining the logistic
fun_log = @(beta, x) ( beta(2)*beta(1) ./ (beta(1) + (beta(2)-beta(1))*exp(-beta(3)*x)) );
% The equation above kinda sucks to interpret.

% This looks good in wolfram alpha:  plot y=-68.46205+(-1.160082 -
-68.46205)/(1+(x/7.776539)^(15.69459))) from 1 to 10
% The equation looks like y = a+(b-a)/(1+(x/c)^d), where a determines the
% final depth, b determines the initial depth, c is where the inflection
% point is, and d is how tightly the curvature is.
```

```
%
% test_fun = @(beta, x) ( beta(1) + (beta(2) - beta(1)) ./ (1 + (x ./ beta(3)) .* exp(beta(4)))
);
%
% guess = [-32, 1, -20, 10];
% newstuff = nlinfit(duckHigh6(:,1), duckHigh6(:,2), test_fun, guess);
% ys = test_fun(newstuff, duckHigh6(:,1));
%
% figure
% plot(duckHigh6(:,1), ys)
% https://math.stackexchange.com/questions/1292750/draw-a-line-output-equation

% Something else, y = a / (1 + b*exp(-(x+c))) could work where a is the max
% depth, b is the steepness, and c is where the inflection point is.
% wolfram alpha:  plot y = -55/(1+5^-(x-1)) from -5 to 5
% https://www.desmos.com/calculator/naf1qogfjn

Duck_init_val = [25,-81,4];
Muk_init_val = [35,-190,14];


duck_newBetaHigh6 = nlinfit(duckHigh6(:,1), duckHigh6(:,2), fun_log, Duck_init_val);
duck_newBetaLow6 = nlinfit(duckLow6(:,1), duckLow6(:,2), fun_log, Duck_init_val);
muk_newBetaHigh3 = nlinfit(mukHigh3(:,1), mukHigh3(:,2), fun_log, Muk_init_val);
muk_newBetaLow3 = nlinfit(mukLow3(:,1), mukLow3(:,2), fun_log, Muk_init_val);

duckLogHigh6 = fun_log(duck_newBetaHigh6, duckHigh6(:,1));
duckLogLow6 = fun_log(duck_newBetaLow6, duckLow6(:,1));
mukLogHigh3 = fun_log(muk_newBetaHigh3, mukHigh3(:,1));
mukLogLow3 = fun_log(muk_newBetaLow3, mukLow3(:,1));



% % As a reference:
% duckdownX5 = duckx(1:5:end);  %every five datapoints
% duckdownX20 = duckx(1:20:end);
% duckdownY5 = ducky(1:5:end);    %need y values as well for matrix size constraints
% duckdownY20 = ducky(1:20:end);
%
% mukdownX5 = mukx(1:5:end);
% mukdownX20 = mukx(1:20:end);
% mukdownY5 = muky(1:5:end);
% mukdownY20 = muky(1:20:end);
%
% duckdx5 = duckdownX5(1):0.01:duckdownX5(end);
% duckspline5 = spline(duckdownX5, duckdownY5, duckdx5);
% duckpchip5 = pchip(duckdownX5, duckdownY5, duckdx5);
%
% mukdx5 = mukdownX5(1):0.01:mukdownX5(end);
% mukspline5 = spline(mukdownX5, mukdownY5, mukdx5);
% mukpchip5 = pchip(mukdownX5, mukdownY5, mukdx5);
%
% duckdx20 = duckdownX20(1):0.01:duckdownX20(end);
% duckspline20 = spline(duckdownX20, duckdownY20, duckdx20);
% duckpchip20 = pchip(duckdownX20, duckdownY20, duckdx20);
%
% mukdx20 = mukdownX20(1):0.01:mukdownX20(end);
% mukspline20 = spline(mukdownX20, mukdownY20, mukdx20);
% mukpchip20 = pchip(mukdownX20, mukdownY20, mukdx20);

%% Area and Error Calculations
```

```
%% Plotting Datasets Together

% figure
% plot(duckDX6, duckSplineHigh6, duckDX6, duckPchipHigh6)
% legend('Spline', 'Pchip')
%
% figure
% plot(duckDX6, duckSplineLow6, duckDX6, duckPchipLow6)
% legend('Spline', 'Pchip')

figure
plot(duckHigh6(:,1), duckLogHigh6, duckLow6(:,1), duckLogLow6, duckNew6(:,1), duckNew6(:,2),
'ro')

figure
plot(mukHigh3(:,1), mukLogHigh3, mukLow3(:,1), mukLogLow3, mukNew3(:,1), mukNew3(:,2), 'ro')
```